

计算机组成原理

计算机组成原理课程组

(刘旭东、高小鹏、肖利民、牛建伟、栾钟治)

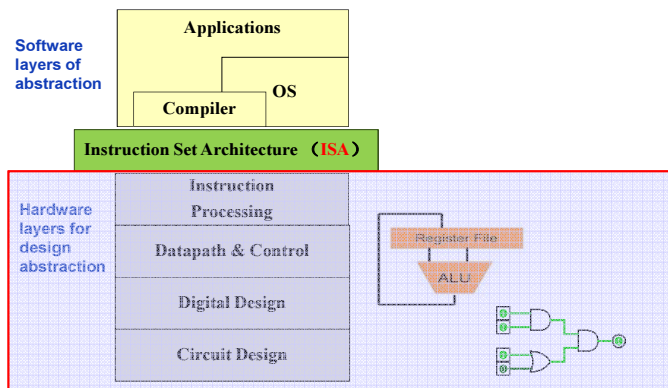
第六讲 MIPS处理器设计

一. 处理器设计概述

1. 处理器的功能与组成
2. 处理器设计的一般方法

二. MIPS模型机

- 三. MIPS单周期处理器设计
- 四. MIPS多周期处理器设计
- 五. MIPS流水线处理器设计



1.1 CPU的功能与组成

❖ CPU的功能：控制指令执行

❖ 指令的四种基本操作

- 取数：读取某主存单元的数据，并传送至某个寄存器；
- 存数：将某个寄存器中的数据存入主存某个单元之中；
- 传送：将某个寄存器中的数据传送至ALU或另一个寄存器；
- 运算：进行某种算术或逻辑运算，结果保存到某个寄存器中。

❖ 指令执行周期（一般性概念）：CPU从指令存储器中读出并执行指令功能的全部时间称为指令周期。包括：

- 取指周期：完成取指令操作和分析指令操作所需时间；
- 取数周期：从数据存储器读出操作数所需时间（包括计算操作数有效地址）；
- 执行周期：完成指令所规定的动作（运算）所需要时间，因指令不同而不同。

1.1 CPU的功能与组成

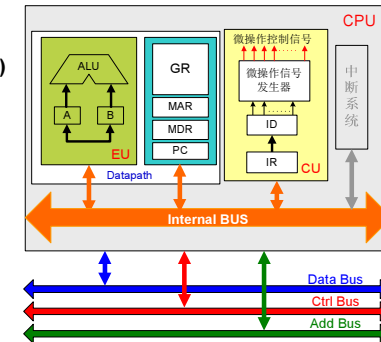
❖CPU所需的功能部件

- 取指令：从存储器中读出指令和分析指令（译码）
 - 指令地址部件：指明当前要读取的指令在存储器中的地址
 - 指令寄存部件：保存从存储器中取来的指令
 - 译码部件：对指令进行译码
- 执行指令：实现指令应该具有的操作功能（包括取数和执行）。
 - 执行部件：ALU、寄存器、数据存储器等等
 - 控制信号逻辑部件：根据指令的操作性质和操作对象的地址（译码结果），在时序信号配合下，产生一系列的微操作控制信号，从而控制计算机的运算器、存储器或输入输出接口等部件工作，实现指令所表示的功能。

1.1 CPU的功能与组成

➤CPU内部结构（内部单总线结构）

- 数据通路 (datapath)
 - ◆ 运算单元
 - ◆ 寄存器单元
- 控制器 (CU)
 - ◆ 指令译码器ID
 - ◆ 控制信号生成
- 内部总线



1.1 CPU的功能与组成

❖CPU的组成

- 执行单元（数据通路，datapath）
 - 运算单元：算术逻辑运算单元（ALU）
 - 寄存器：通用寄存器组（GPRs），标志寄存器（FR，又称程序状态字PSW），临时寄存器（TR）
- 控制单元（控制器，control）：
 - 指令地址部件：程序计数器（PC — Program Counter）
 - 指令寄存部件：指令寄存器（IR — Instruction Register）
 - 译码部件：指令译码器（ID — Instruction Decoder）
 - 微操作控制信号产生部件：产生计算机其他部件所需要的所有微操作控制信号，有组合逻辑和微程序等实现方式。
 - 时序部件：产生时序信号

1.1 CPU的功能与组成

❖数据通路

- 指令执行过程中，指令数据流所经过的部件和路径总称，用以实现数据的传送、处理和存储等功能，是**指令的执行部件**。

➤构成

- 组合逻辑元件（操作元件）：ALU、译码器、多路选择器等
- 存储元件（状态元件）：存储器、寄存器等

➤部件间连接方式

- 总线连接方式（CPU内部总线）
- 分散连接方式

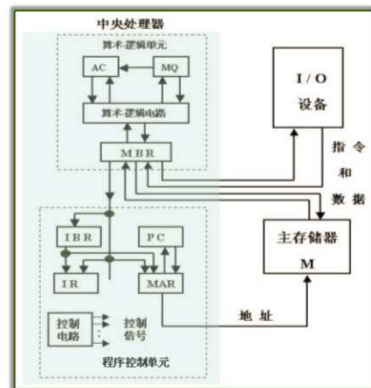
❖控制器

- 对指令进行译码并生成指令执行所需的控制信号，以实现对数据通路中各件的功能控制，以及相应路径的开关控制等，是**指令的控制部件**。

1.1 CPU的功能与组成

❖ 简单的数据通路示例

- 取指令路径
 - PC→MAR
 - Read Mem.
 - M→MBR→IBR→IR
- 取操作数的路径
 - 操作数地址→MAR
 - Read Mem.
 - M→MBR→ALU
- 运算结果保存路径
 - ALU结果→MBR
 - 结果地址→MAR
 - Write Mem.

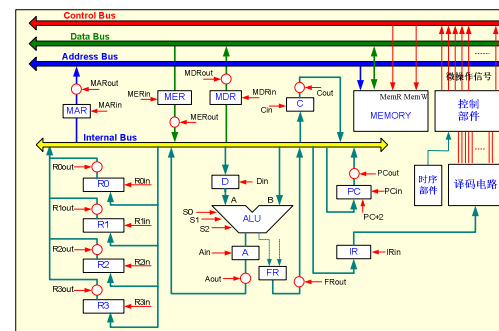


早期累加器型数据通路

1.1 CPU的功能与组成

❖ 单总线数据通路示例

- | 取指令路径 | 取操作数的路径 | 运算结果保存路径 |
|--|---|---|
| <ul style="list-style-type: none"> PC→IB→MAR MemR M→MER→IB→IR | <ul style="list-style-type: none"> 地址→IB→MAR MemR M→MER→IB→ALU | <ul style="list-style-type: none"> ALU结果→IB→MDR 结果地址→IB→MAR MemW |



单总线数据通路

1.1 CPU的功能与组成

- ❖ 指令功能的形式化描述: RTL (Register Transfer Language, 寄存器传送语言)

- \leftarrow : 数据传送方向;
- $R[a]$: 寄存器 a ;
- $M[a]$: 主存中地址为 a 的单元;
- PC : 程序计数器
- $f(data)$: 表示对数据 $data$ 进行 f 操作

◆ 示例

- $R[c] \leftarrow R[a] + R[b]$ // 寄存器a加寄存器b的结果送寄存器c
- $R[c] \leftarrow R[a] \text{ op } R[b]$ // 寄存器a与寄存器b进行op运算结果送寄存器c
- $\text{Signext}(\text{imm}16)$ // 对数imm16进行Signext（符号扩展）运算
- $R[a] \leftarrow M[b]$ // 取数操作，读取主存单元b的数据传送至寄存器a
- $M[a] \leftarrow R[b]$ // 存数操作，将寄存器b中的数据写入主存单元a中

1.3 处理器设计的一般方法

◆ 设计步骤

1. 分析指令系统需求：包括指令格式、指令类型、每种指令的功能、寻址方式等；
2. 数据通路构建
 - ① 根据指令需求选择数据通路部件，如PC、ALU、寄存器堆、指令/数据存储器、多路开关等等；
 - ② 根据指令执行流程构建每种类型指令的数据通路；
 - ③ 对所有类型指令执行数据通路综合形成综合数据通路。
3. 控制器设计
 - ① 确定控制器时序控制方式（单周期、或多周期或其他）
 - ② 根据每种类型指令执行流程，确定该执行执行时各个数据通路部件所需要的控制信号与相应状态、条件；
 - ③ 对控制信号进行综合以得到每个控制信号的逻辑方程；
 - ④ 逻辑电路实现各个控制信号。

第六讲 MIPS处理器设计

一. 处理器设计概述

1. 处理器的功能与组成
2. 处理器设计的一般方法

二. MIPS模型机

1. MIPS模型机指令集
2. MIPS模型机数据通路部件
3. 时钟同步方法

三. MIPS单周期处理器设计

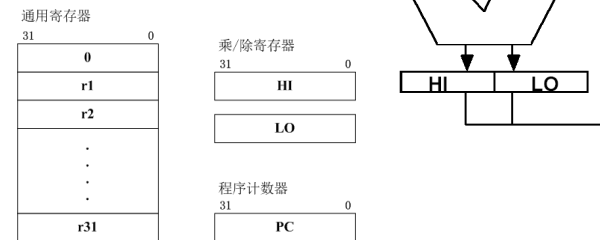
四. MIPS多周期处理器设计

五. MIPS流水线处理器设计

2.1 MIPS模型机指令集

❖ MIPS 寄存器结构

- 32位虚拟地址空间
- 32个32位GPRs (通用寄存器)
- 32个32位FPRs (浮点数寄存器)
- HI, LO, PC



2.1 MIPS模型机指令集

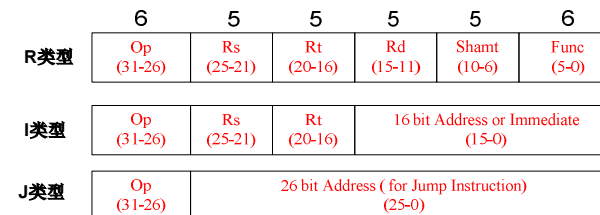
MIPS 模型机寄存器使用约定

寄存器编号	寄存器名称	用途
0	\$zero	常数0
1	\$at	保留给汇编器使用
2 ~ 3	\$v0 ~ \$v1	结果值和表达式求值
4 ~ 7	\$a0 ~ \$a3	参数
8 ~ 15	\$t0 ~ \$t7	临时变量
16 ~ 23	\$s0 ~ \$s7	数据寄存器
24 ~ 25	\$t8 ~ \$t9	其他临时变量
26 ~ 27	\$k0 ~ \$k1	保留给操作系统使用
28	\$gp	全局指针
29	\$sp	栈指针
30	\$fp	帧指针
31	\$ra	返回地址

2.1 MIPS模型机指令集

❖ MIPS 指令格式

- Op: 6 bits, Opcode
- Rs: 5 bits, The first register source operand
- Rt: 5 bits, The second register source operand
- Rd: 5 bits, The register destination operand
- Shamt: 5 bits, Shift amount (shift instruction)
- Func: 6 bits, function code (another Opcode)

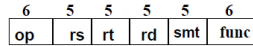


2.1 MIPS模型机指令集

❖ MIPS 寻址方式

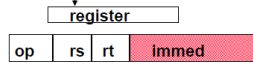
R-format:

Register (direct)

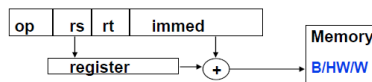


I-format:

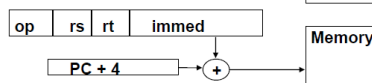
Immediate



Base或Index



PC-relative



J-format:

Pseudodirect



2.1 MIPS模型机指令集

模型机指令集 (8条指令)

指令格式	指令	功能	说明
R 类型	add rd, rs, rt	$R[rd] \leftarrow R[rs] + R[rt]$	加运算 : 寄存器 rs 和寄存器 rt 相加, 结果送寄存器 rd
	sub rd, rs, rt	$R[rd] \leftarrow R[rs] - R[rt]$	减运算 : 寄存器 rs 和寄存器 rt 相减, 结果送寄存器 rd
	and rd, rs, rt	$R[rd] \leftarrow R[rs] \& R[rt]$	与运算 : 寄存器 rs 和寄存器 rt 按位与, 结果送寄存器 rd
	or rd, rs, rt	$R[rd] \leftarrow R[rs] R[rt]$	或运算 : 寄存器 rs 和寄存器 rt 按位或, 结果送寄存器 rd
I 类型	lw rt, rs, imm16	$Add = R[rs] + \text{Signext}(imm16)$ $R[rt] \leftarrow M[Add]$	取字 : 寄存器 rs 和立即数 imm16 (符号扩展至 32 位) 相加得到内存地址, 从内存该地址单元读取数据送 rt
	sw rt, rs, imm16	$Add = R[rs] + \text{Signext}(imm16)$ $M[Add] \leftarrow R[rt]$	存字 : 寄存器 rs 和立即数 imm16 (符号扩展至 32 位) 相加得到内存地址, 寄存器 rt 数据写入内存该地址单元
	beq rs, rt, imm16	If $(R[rs] - R[rt] = 0)$ then $PC \leftarrow PC + \text{Signext}(imm16) < 2$	分支 : 如果寄存器 rs 与 rt 相等, 则转移 (imm16 符号扩展至 32 位), 否则顺序执行。(取指令后, PC+4)
J 类型	j target	$PC(31:2) \leftarrow PC(31:28) \text{target}(25:0)$	跳转 : 当前 PC 的高 4 位与 target (26 位) 拼接成 30 位目标地址送 PC (31:2)。(取指令后, PC+4)。

2.1 MIPS模型机指令集

模型机指令编码

R 类型格式	OP (31~26)	Rs (25~21)	Rt (20~16)	Rd (15~11)	Shamt (10~6)	Funct (5~0)
add rd, rs, rt	000000	rs	rt	rd	XXXXX	100000
sub rd, rs, rt	000000	rs	rt	rd	XXXXX	100010
and rd, rs, rt	000000	rs	rt	rd	XXXXX	100100
or rd, rs, rt	000000	rs	rt	rd	XXXXX	100101

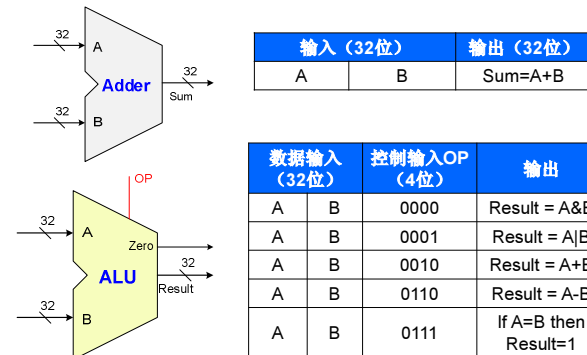
I 类型格式	OP (31~26)	Rs (25~21)	Rt (20~16)	16 bits immediate or address (15~0)
lw rt, rs, imm16	100011	rs	rt	imm16
sw rt, rs, imm16	101011	rs	rt	imm16
beq rs, rt, imm16	000100	rs	rt	imm16

J 类型格式	OP (31~26)	26 bits address
j target	000010	target

2.2 数据通路部件

❖ 组合部件:

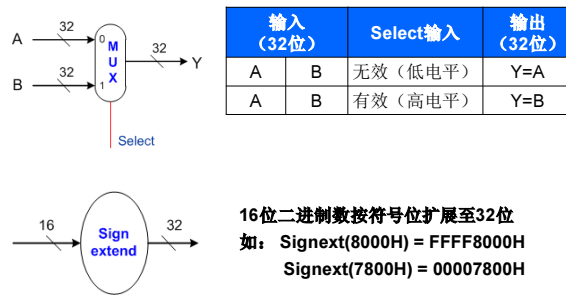
➤ 加法器 (Adder)、算术逻辑运算单元 (ALU)



2.2 数据通路部件

❖ 组合部件

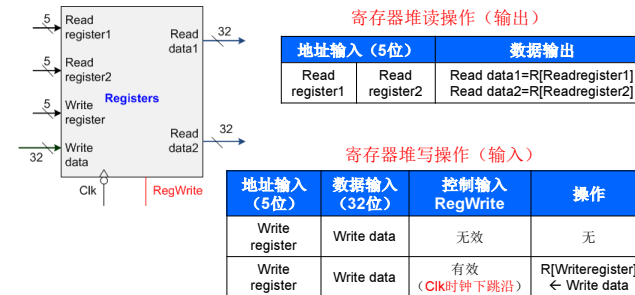
- 多路选择器 (Mux)、符号扩展器 (Signext)
- 多路选择器有二选一，三选一，四选一等。



2.2 数据通路部件

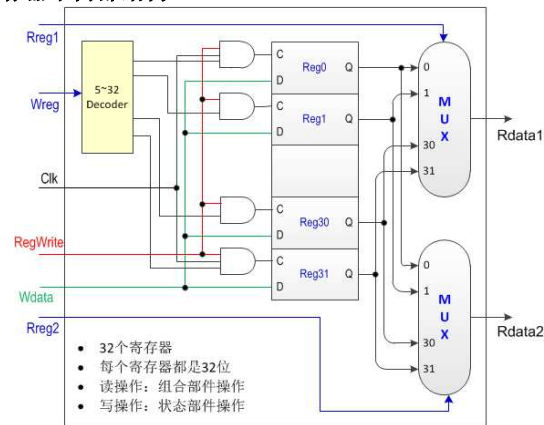
❖ 寄存器堆 (32个寄存器)

- 两个32位输出端口
- 一个32位输入端口



2.2 数据通路部件

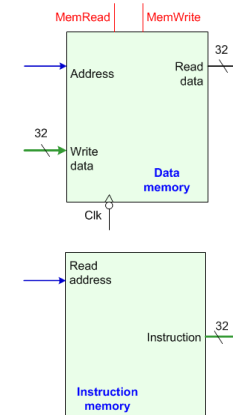
❖ 寄存器堆内部结构



2.2 数据通路部件

❖ 数据存储器DM (理想存储器)

- 单输入总线: Write data
- 单输出总线: Read data
- 读: MemRead控制信号有效时, 地址线(Address)选择的存储字被放在Read data输出总线上 (组合元件操作)
- 写: MemWrite控制信号有效且时钟信号Clk下跳沿时, Write data总线上的数据被写入地址选择的存储单元中 (状态元件操作)



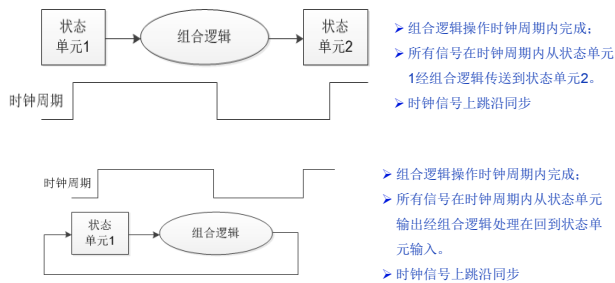
❖ 指令存储器IM

- 指令地址(Read address)选择的指令被放在Instruction输出线上 (组合元件操作)

2.3 时钟同步方法

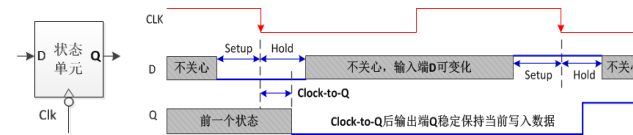
❖ 时钟同步方法

- 以时钟周期信号为基准，确定数据读出和写入的时刻。
- 采用边沿触发的时钟同步方法，如下跳沿触发，意味着所有状态元件（寄存器、存储器）的数据写入都发生在时钟周期的下跳沿时刻。



2.3 时钟同步方法

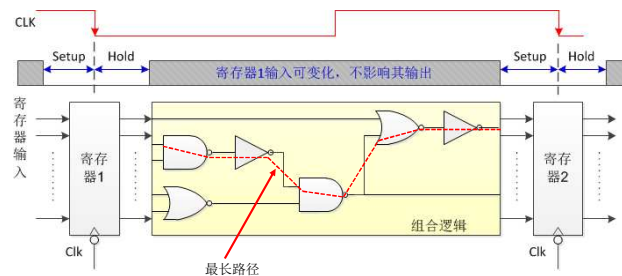
❖ 状态单元的时序



- 建立时间 (Setup Time)：触发时钟边沿之前输入必须稳定的时间；
- 保持时间 (Hold Time)：触发时钟边沿之后输入仍需稳定的时间；
- Clock-to-Q：从触发时钟边沿到输出有效的时间。

2.3 时钟同步方法

❖ 时钟周期



- 数据通路由“... + 状态元件 + 操作元件(组合电路) + 状态元件 + ...”组成
- 状态元件存储信息，所有操作元件从状态单元接收输入，并将输出写入状态单元中。其输入为前一时钟生成的数据，输出为当前时钟所用的数据
- $\text{Cycle Time} = \text{Clock-to-Q} + \text{Longest Delay Path} + \text{Setup} + \text{Clock Skew}$

第六讲 MIPS处理器设计

一. 处理器设计概述

1. 处理器的功能与组成
2. 处理器设计的一般方法

二. MIPS模型机

三. MIPS单周期处理器设计

1. 单周期数据通路设计
2. 单周期控制器设计
3. 单周期性能分析

四. MIPS多周期处理器设计

五. MIPS流水线处理器设计

3.1 单周期数据通路设计

❖ 单周期

- 所有指令执行周期固定为单一时钟周期， $CPI=1$ 。

❖ 通路设计考虑

- 哈佛体系结构：使用指令存储区（IM）和数据存储区（DM）分别保存指令和数据
- 先为每类指令设计独立的数据通路，然后再考虑数据通路合并。

❖ 指令执行的共性

- 根据PC从指令存储器读取指令，取指令后， $PC+4$ ；
- 模型机7条指令在读取寄存器后，都要使用ALU
 - LW/SW（存储访问）指令用ALU计算数据地址
 - ADD/SUB/AND/OR（算术逻辑）指令用ALU完成算术逻辑运算
 - BEQ（分支）指令用ALU进行比较（减法运算）

3.1 单周期数据通路设计

❖ 分析指令执行步骤，确定数据通路所需部件和部件间连接

- 模型机指令执行过程一般会分为如下几个步骤：
 - 取指令：根据PC访问指令存储器获得指令，然后 $PC+4$ ；
 - 读寄存器：根据指令格式读取相应寄存器操作数
 - ALU运算：在ALU完成相应的算术逻辑运算
 - 数据存取：LW/SW指令的数据存储器访问
 - 写寄存器：运算类指令和LW指令要把数据写入寄存器

❖ 使用数据通路设计表格

- 表格记录数据通路部件输入端的输入来源
- 暂不考虑控制信号

指令	Adder		PC	IM Add.	Registers				ALU		DM	
	A	B			Reg1	Reg2	Wreg	Wdata	A	B	Add.	Wdata

3.1 单周期数据通路设计——取指与PC自增

1. 取指和PC自增数据通路（所有指令）

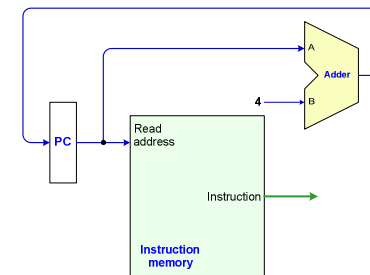
- 功能描述
 - 取指：IM Address $\leftarrow PC$, instruction = IM[PC]
 - PC自增：PC $\leftarrow PC + 4$
- 所需部件：PC，Adder（实现PC加4），指令存储器IM

指令	Adder		PC	IM Add.	Registers				ALU		DM	
	A	B			Reg1	Reg2	Wreg	Wdata	A	B	Add.	Wdata
R型指令	PC	4	Adder	PC								
Lw	PC	4	Adder	PC								
Sw	PC	4	Adder	PC								
Beq	PC	4	Adder	PC								

3.1 单周期数据通路设计——取指与PC自增

1. 取指和PC自增数据通路图

- 功能描述
 - 取指：instruction = IM[PC]
 - PC自增：PC $\leftarrow PC + 4$
- 所需部件：PC，Adder（实现PC加4），IM



3.1 单周期数据通路设计——R型指令数据通路

2. R型指令数据通路 (add,sub,and,or指令, 以add为例)

➤ add rd, rs, rt

➤ 功能描述

$$R[rd] \leftarrow R[rs] + R[rt]$$

➤ 通路部件: 寄存器堆、ALU

指令	Adder		PC	IM Add.	Registers				ALU		DM	
	A	B			Reg1	Reg2	Wreg	Wdata	A	B	Add.	Wdata
R型指令	PC	4	Adder	PC	Rs	Rt	Rd	ALU	Rdata1	Rdata2		
Lw	PC	4	Adder	PC								
Sw	PC	4	Adder	PC								
Beq	PC	4	Adder	PC								

3.1 单周期数据通路设计——R型指令数据通路

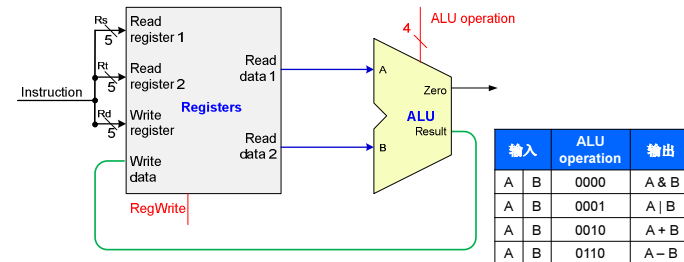
2. R型指令数据通路 (add,sub,and,or指令, 以add为例)

➤ add rd, rs, rt

➤ 功能描述

$$R[rd] \leftarrow R[rs] + R[rt]$$

➤ 通路部件: 寄存器堆、ALU



3.1 单周期数据通路设计——LW指令数据通路

3. 取数指令 (lw) 数据通路

➤ lw rt, rs, imm16

➤ 功能描述:

$$R[rt] \leftarrow DM[R[rs] + \text{Signext}(imm16)]$$

➤ 通路部件: 寄存器堆, ALU, 符号扩展单元Signext, 数据存储器DM

指令	Adder		PC	IM Add.	Registers				ALU		DM		Sign-ext
	A	B			Reg1	Reg2	Wreg	Wdata	A	B	Add.	Wdata	
R型指令	PC	4	Adder	PC	Rs	Rt	Rd	ALU	Rdata1	Rdata2			
Lw	PC	4	Adder	PC	Rs		Rt	DM	Rdata1	Sign-ext	ALU		imm16
Sw	PC	4	Adder	PC									
Beq	PC	4	Adder	PC									

3.1 单周期数据通路设计——LW指令数据通路

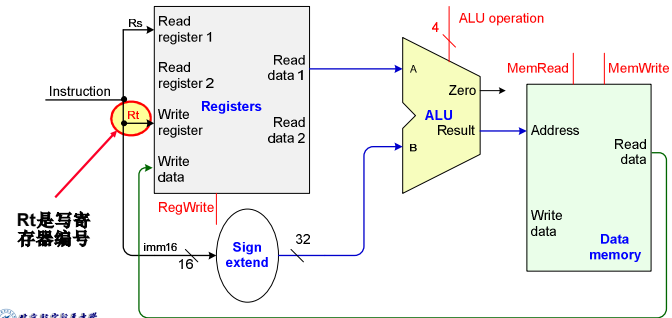
3. 取数指令 (lw) 数据通路

➤ lw rt, rs, imm16

➤ 功能描述:

$$R[rt] \leftarrow DM[R[rs] + \text{Signext}(imm16)]$$

➤ 通路部件: 寄存器堆, ALU, 符号扩展单元Signext, 数据存储器DM



3.1 单周期数据通路设计——SW指令数据通路

4. 存数指令 (sw) 数据通路

➤ sw rt, rs, imm16

Op (31-26)	Rs (25-21)	Rt (20-16)	16 bit Address or Immediate (15-0)
---------------	---------------	---------------	---------------------------------------

➤ 功能描述:

▪ $DM[R[rs] + \text{Signext}(imm16)] \leftarrow R[rt]$

➤ 通路部件: 寄存器堆, ALU, 符号扩展单元Signext, 数据存储器DM

指令	Adder		PC	IM Add.	Registers				ALU		DM		Sign-ext
	A	B			Reg1	Reg2	Wreg	Wdata	A	B	Add.	Wdata	
R型指令	PC	4	Adder	PC	Rs	Rt	Rd	ALU	Rdata1	Rdata2			
Lw	PC	4	Adder	PC	Rs		Rt	DM	Rdata1	Signext	ALU		imm16
Sw	PC	4	Adder	PC	Rs	Rt			Rdata1	Signext	ALU	Rdata2	imm16
Beq	PC	4	Adder	PC									

3.1 单周期数据通路设计——SW指令数据通路

4. 存数指令 (sw) 数据通路

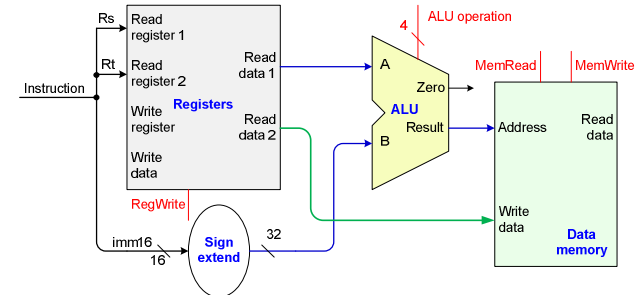
➤ sw rt, rs, imm16

Op (31-26)	Rs (25-21)	Rt (20-16)	16 bit Address or Immediate (15-0)
---------------	---------------	---------------	---------------------------------------

➤ 功能描述:

▪ $DM[R[rs] + \text{Signext}(imm16)] \leftarrow R[rt]$

➤ 通路部件: 寄存器堆, ALU, 符号扩展单元Signext, 数据存储器DM



3.1 单周期数据通路设计——R型指令与访存指令通路合并

5. R型指令与访存指令数据通路合并: 增加多路选择Mux

Op (31-26)	Rs (25-21)	Rt (20-16)	Rd (15-11)	Shamt (10-6)	Func (5-0)
---------------	---------------	---------------	---------------	-----------------	---------------

R型指令格式

Op (31-26)	Rs (25-21)	Rt (20-16)	16 bit Address or Immediate (15-0)
---------------	---------------	---------------	---------------------------------------

LW/SW指令格式

指令	Adder		PC	IM Add.	Registers				ALU		DM		Sign-ext
	A	B			Reg1	Reg2	Wreg	Wdata	A	B	Add.	Wdata	
R型指令	PC	4	Adder	PC	Rs	Rt	Rd	ALU	Rdata1	Rdata2			
Lw	PC	4	Adder	PC	Rs		Rt	DM	Rdata1	Signext	ALU		imm16
Sw	PC	4	Adder	PC	Rs	Rt			Rdata1	Signext	ALU	Rdata2	imm16
合并	PC	4	Adder	PC	Rs	Rt	Rd	ALU	Rdata1	Signext	ALU	Rdata2	imm16

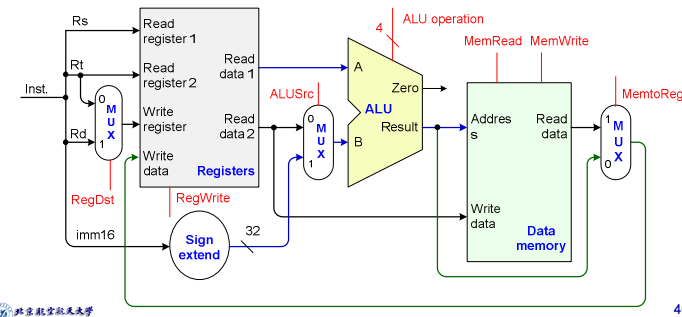
输入端数据源出现多个选择, 需要加入多路选择器MUX

3.1 单周期数据通路设计——R型指令与访存指令通路合并

5. R型指令与访存指令数据通路合并

➤ 增加3个二选一多路选择器MUX

- 寄存器堆写入端地址选择MUX, 选择控制信号 RegDst
- ALU输入端B数据源选择MUX, 选择控制信号 ALUSrc
- 寄存器堆写入端数据源选择MUX, 选择控制信号 MemtoReg



3.1 MIPS的数据通路设计——Beq指令数据通路

6. 分支指令数据通路

➤ beq rs, rt, imm16

➤ 功能描述:

▪ If (R[rs] - R[rt] = 0) then $PC \leftarrow (PC + 4) + \text{Signext}(\text{imm16}) \ll 2$
else $PC \leftarrow PC + 4$

➤ 通路部件: 寄存器堆, ALU, 增加一加法器Nadd, 符号扩展Signext, 移位器

指令	Adder		PC	IM Add.	Registers				ALU		DM	Sign-ext	Nadd	
	A	B			Reg1	Reg2	Wreg	Wdata	A	B				
R型与访存	PC	4	Adder	PC	Rs	Rt	Rd Rt	ALU DM	Rdata1	Rdata2 Sign-ext	ALU	Rdata2	imm16	
Beq	PC	4	Adder Nadd	PC	Rs	Rt			Rdata1	Rdata2			imm16	Adder Shift

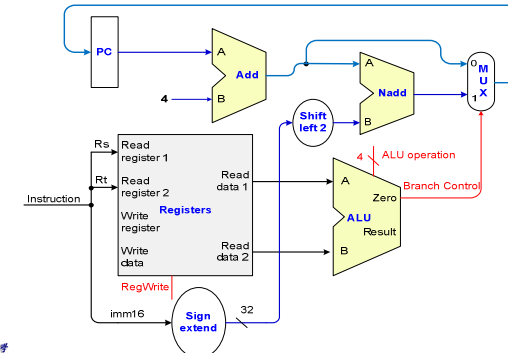
需要一个MUX, ALU的判零输出端Zero可直接作为该MUX的选择控制

3.1 MIPS的数据通路设计——Beq指令数据通路

6. 分支指令数据通路

➤ beq rs, rt, imm16

➤ 通路部件: 寄存器堆, ALU, 增加一Adder, 符号扩展Signext, 移位器



3.1 单周期数据通路设计

7. MIPS数据通路再合并

➤ 支持: R类型指令、内存访问指令 (lw/sw)、beq指令

指令	Adder		PC	IM Add.	Registers				ALU		DM	Sign-ext	Nadd	
	A	B			Reg1	Reg2	Wreg	Wdata	A	B				
R型与访存	PC	4	Adder	PC	Rs	Rt	Rd Rt	ALU DM	Rdata1	Rdata2 Sign-ext	ALU	Rdata2	imm16	
Beq	PC	4	Adder Nadd	PC	Rs	Rt			Rdata1	Rdata2			imm16	Adder Shift
合并	PC	4	Adder Nadd	PC	Rs	Rt	Rd Rt	ALU DM	Rdata1	Rdata2 Sign-ext	ALU	Rdata2	imm16	Adder Shift

需要4个二选一多路选择器MUX

- PC输入端数据源选择MUX, 选择控制信号 PCSrc
- 寄存器堆写入端地址选择MUX, 选择控制信号 RegDst
- ALU输入端B数据源选择MUX, 选择控制信号 ALUSrc
- 寄存器堆写入端数据源选择MUX, 选择控制信号 MemtoReg

第六讲 MIPS处理器设计

一. 处理器设计概述

1. 处理器的功能与组成
2. 处理器设计的一般方法

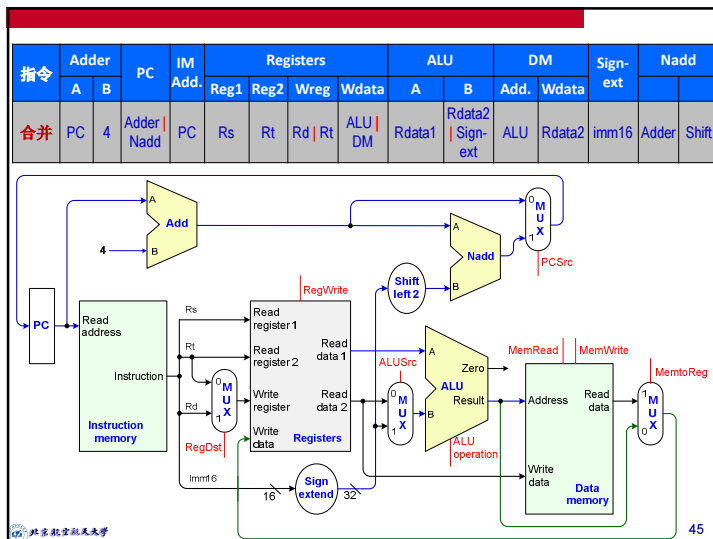
二. MIPS模型机

三. MIPS单周期处理器设计

1. 单周期数据通路设计
2. 单周期控制器设计
3. 单周期性能分析

四. MIPS多周期处理器设计

五. MIPS流水线处理器设计



3.2 单周期控制器设计

❖ 单周期通路所需控制信号

➢ ALU控制 (ALU Operation) : 4位

➢ 其他控制信号: 7个

ALU 控制

输入	ALU operation	ALU 运算
A B	0000	A & B
A B	0001	A B
A B	0010	A + B
A B	0110	A - B

7个控制信号

控制信号	失效时作用	有效时作用
RegDst	寄存器堆写入端地址来选择Rt字段	寄存器堆写入端地址选择Rd字段
RegWrite	无	把数据写入寄存器堆中对应寄存器
ALUSrc	ALU输入端B选择寄存器堆输出R[rt]	ALU输入端B选择Signext输出
PCSrc	PC输入源选择PC+4	PC输入选择beq指令的目的地址
MemRead	无	数据存储器DM读数据 (输出)
MemWrite	无	数据存储器DM写数据 (输入)
MemoReg	寄存器堆写入端数据来自ALU输出	寄存器堆写入端数据来自DM输出

3.2 单周期控制器设计

❖ 控制器分成两部分：主控单元和ALU控制单元

➢ 主控单元

▪ 输入：指令操作码字段 Op (指令31:26位)

▪ 输出：

- 7个控制信号
- ALU控制单元所需的2位输入ALUOp

➢ ALU控制单元

▪ 输入：

- 主控单元生成的ALUOp
- 功能码字段Func (指令5:0位)

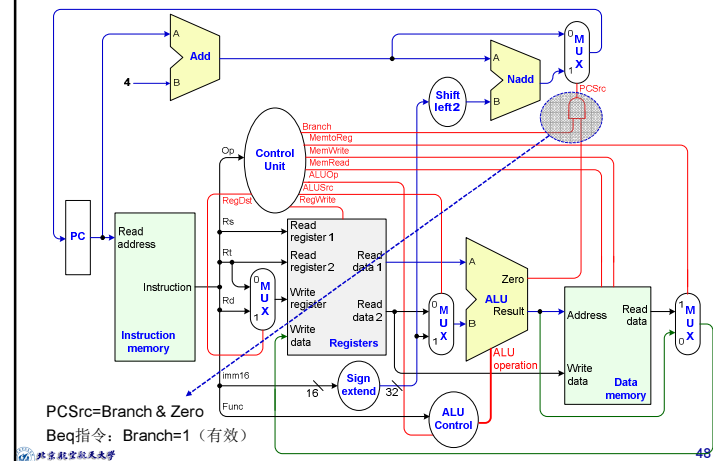
▪ 输出：ALU运算控制信号 ALU operation (4位)

ALUOp指明ALU的运算类型

- 00: 访存指令所需加法
- 01: beq指令所需减法
- 10: R型指令功能码决定

Op (31-26)	Rs (25-21)	Rt (20-16)	Rd (15-11)	Shamt (10-6)	Func (5-0)
---------------	---------------	---------------	---------------	-----------------	---------------

3.2 单周期控制器设计——模型机数据通路 (带控制单元)



3.2 单周期控制器设计

❖ 主控单元控制信号分析

➢ RegDst

- R型指令: **RegDst=1**, 选择Rd
- Lw指令: **RegDst=0**, 选择Rt
- 其他指令: 不关心

➢ ALUSrc

- R型指令: **ALUSrc=0**, 选择寄存器堆的 Read data2 输出
- Lw指令: **ALUSrc=1**, 选择Signext的输出
- Sw指令: **ALUSrc=1**, 选择Signext的输出
- Beq指令 (减法运算): **ALUSrc=0**, 选择 Read data2 输出

➢ MemtoReg

- R型指令: **MemtoReg=0**, 选择 ALU 输出
- Lw指令: **MemtoReg=1**, 选择数据存储器DM输出
- 其他指令: 不关心

➢ Branch

- Beq指令: **Branch=1**, 此时若Zero=1, PC输入选择加法器Nadd输出 (分支指令目的地址), 否则选择加法器Add输出 (PC+4)
- 其他指令: **Branch=0**, PC输入选择加法器Add输出 (PC+4)

3.2 单周期控制器设计

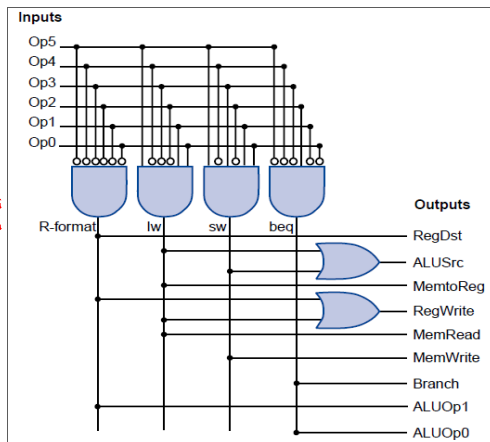
主控单元真值表

Input or output	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

提供给ALU控制单元

3.2 单周期控制器设计

主控单元
逻辑实现



3.2 单周期控制器设计

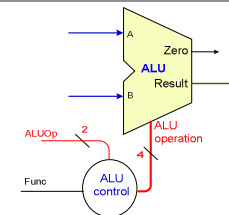
❖ ALU控制单元

➢ 输入:

- 指令的Func字段 (指令5:0位)
- 由主控单元生成的 **ALUOp**

➢ ALUOp指明ALU的运算类型

- 00: 访存指令所需的加法
- 01: beq指令所需的减法
- 10: R型指令功能码字段决定

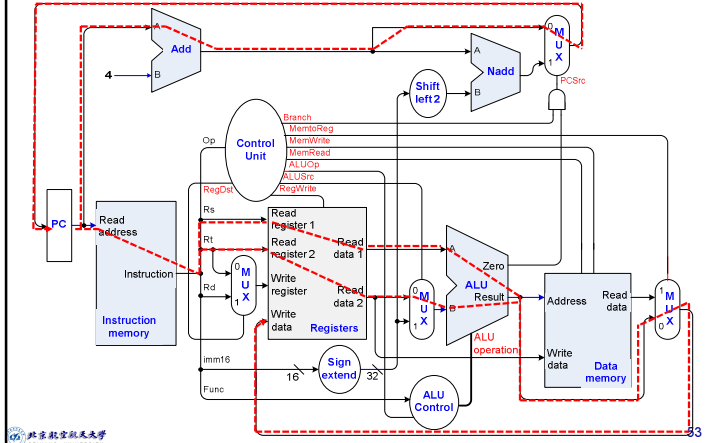


指令	Func字段	ALUOp	ALU运算类型	ALU operation
Lw	XXXXXX	00	加	0010
Sw	XXXXXX	00	加	0010
Beq	XXXXXX	01	减	0110
Add	100 000	10	加	0010
Sub	100 010	10	减	0110
And	100 100	10	与	0000
Or	100 101	10	或	0001

ALU控制单元真值表

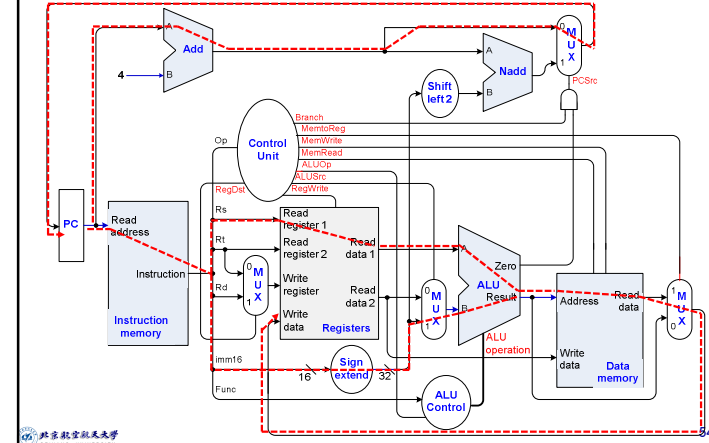
3.2 单周期控制器设计

❖ R型指令执行数据流示意



3.2 单周期控制器设计

❖ LW指令执行数据流示意



3.2 单周期控制器设计

❖ MIPS数据通路（扩展实现跳转指令j）

➢ j add26

➢ 功能描述

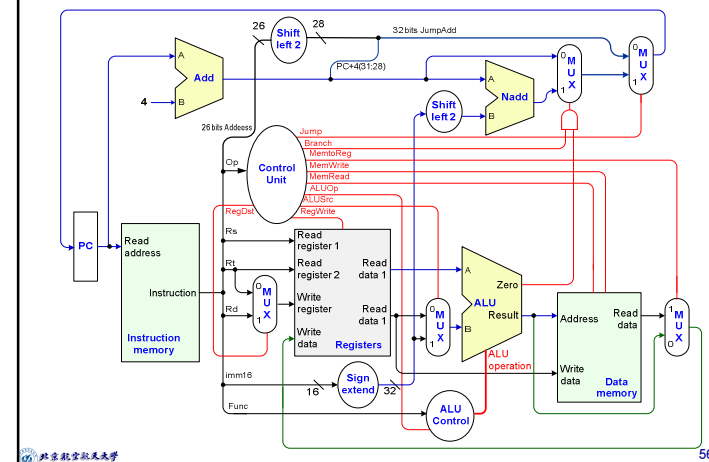
■ $PC \leftarrow PC + 4[31:28] \parallel \text{add26} < 2$

➢ 功能部件：Adder, 移位器

指令	Adder		PC	IM Ad. d.	Registers				ALU		DM		Sign- ext	Nadd	
	A	B			Reg1	Reg2	Wreg	Wdata	A	B	Add.	Wdata			
合并	PC	4	Adder Nadd	PC	Rs	Rt	Rd Rt	ALU DM	Rdata1	Rdata2 Sign- ext	ALU	Rdata2	imm16	Adder	Shift
J指令	PC	4	jumpadd												

PC输入数据源又多了一个选择，增加一个二选一MUX

3.2 单周期控制器设计（包含跳转指令的数据通路）



小结：数据通路设计的一般性方法

单指令 数据通路 构造

```
for each 指令
  for each 新增需求
    case 可以合并至已有部件:
      修改部件设计描述、HDL建模: {F', I', O'}
    case 需要新增部件:
      建立新部件设计描述、HDL建模: {F, I, O}
      增加新部件
  for each 部件
    设置输入来源
```

多数据通路 路综合

按垂直方向合并数据通路，并去除相同项
for each 输入来源多于1个的输入端
部署1个MUX (MUX的输入规模为输入来源数)
MUX设计定义、HDL建模

系统实现

HDL建模：连接所有的部件及所有的MUX

部件描述示例——PC

4.1.1. 基本描述

PC 模块的主要功能是将 NPC[31:0] 的值保存并输出。PC 的各种取值将根据所执行的指令、外部状态(中断)及处理器控制器的当前状态的不同，由数据通路其他部件生成。

4.1.2. 模块接口

表 4-1 PC 接口信号定义

信号名	方向	描述
Clk	I	MIPS-C 处理器时钟
Reset	I	复位信号
NextPC[31:0]	I	下一个 PC 值
PCWr	I	PC 写使能
PC[31:0]	O	PC 输出

4.1.3. 功能定义

PC 模块的核心是一个寄存器。该寄存器在 PCWr 有效时将 NextPC[31:0] 锁存并输出。

表 4-2 PC 功能需求定义

编号	功能名称	功能描述
1	初始化	当 Reset 信号有效后，PC 输出 0xBF000000。
2	PC 更新	当时钟上升沿到来时，PCWr 有效则将 NPC 写入 PC 内部，并且从 PC 端口输出。

部件HDL建模示例——PC

```
16 `timescale 1ns/1ns
17
18 module PC( CLK_I, Reset_I, Addr_I, PCWrite_I, PC_O );
19   input      CLK_I;           // system clock
20   input      Reset_I;         // reset signal
21   input [31:0] Addr_I;        // next PC
22   input      PCWrite_I;       // write enable
23   output [31:0] PC_O;         // PC output
24
25   /* internal reg and wire */
26   reg [31:0] addr;             // latch the address
27
28   /* read register */
29   assign PC_O = addr;
30
31   always@(posedge CLK_I or posedge Reset_I)
32   begin
33     if(Reset_I)
34       addr <= 'hBF000000;
35     else if( PCWrite_I )
36       addr <= Addr_I;
37   end
38
39
40 endmodule
```

第六讲 MIPS处理器设计

一. 处理器设计概述

1. 处理器的功能与组成
2. 处理器设计的一般方法

二. MIPS模型机

三. MIPS单周期处理器设计

1. 单周期数据通路设计
2. 单周期控制器设计

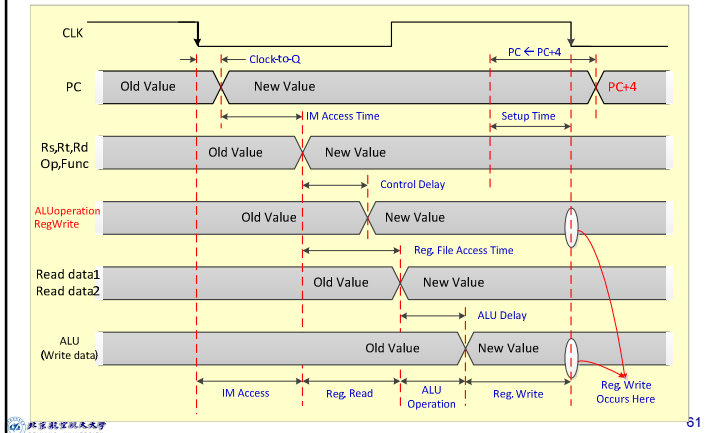
3. 单周期性能分析

四. MIPS多周期处理器设计

五. MIPS流水线处理器设计

3.3 单周期数据通路性能分析

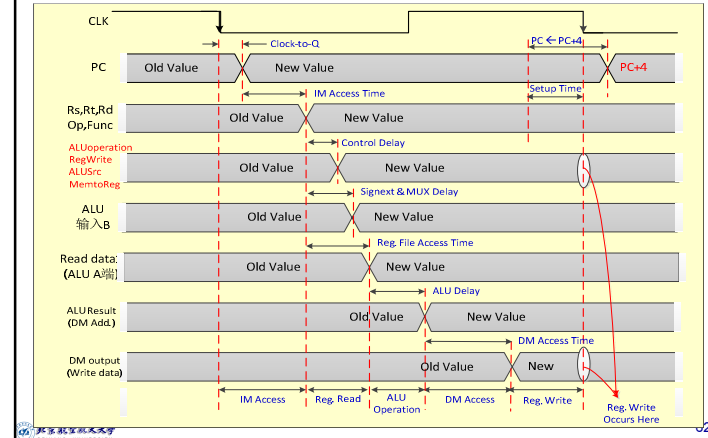
❖ R型指令的指令周期



61

3.3 单周期数据通路性能分析

❖ LW指令的指令周期



62

3.3 单周期数据通路性能分析

❖ 指令周期（指令执行时间）

➢ R指令周期

- 取指令（IM Access Time）
- 读寄存器（Register File Access Time）
- ALU运算（ALU Operation）
- 写寄存器（Register File Access Time）

➢ LW指令周期

- 取指令（IM Access Time）
- 读寄存器（Register Access Time）
- ALU运算（ALU Operation）
- 读数据（DM Access Time）
- 写寄存器（Register File Access Time）

实际上，不同类型的指令可能具有不同的指令周期

3.3 单周期数据通路性能分析

MIPS不同类型指令的指令周期

Instruction class	Functional units used by the instruction class				
R-type	Instruction fetch	Register access	ALU	Register access	
Load word	Instruction fetch	Register access	ALU	Memory access	Register access
Store word	Instruction fetch	Register access	ALU	Memory access	
Branch	Instruction fetch	Register access	ALU		
Jump	Instruction fetch				

数据通路各部分以及各类指令的执行时间

Instruction class	Instruction memory	Register read	ALU operation	Data memory	Register write	Total
R-type	200	50	100	0	50	400 ps
Load word	200	50	100	200	50	600 ps
Store word	200	50	100	200		550 ps
Branch	200	50	100	0		350 ps
Jump	200					200 ps

3.3 单周期数据通路性能分析

❖ 指令执行时间计算

1. 方式一：采用单周期，即所有指令周期固定为单一时钟周期

■ 时钟周期有最长的指令决定（LW指令），为 **600ps**

■ 指令平均周期 = **600ps**

2. 方式二：不同类型指令采用不同指令周期（可变时钟周期）

■ 假设指令在程序中出现的频率

– lw指令 : 25%

– sw指令 : 10%

– R类型指令 : 45%

– beq指令 : 15%

– j指令 : 5%

■ 平均指令执行时间

$$600 \times 25\% + 550 \times 10\% + 400 \times 45\% + 350 \times 15\% + 200 \times 5\% = 447.5\text{ps}$$

— 若采用可变时钟周期，时间性能比单周期更高；

— 但控制比单周期要复杂、困难，得不偿失。

— 改进方法：改变每种指令类型所用的时钟数，即采用多周期实现